A GNN approach for events-characters linking across documents in TTRPGs

Leonardo Bisazza, Fabrizio Casadei

bisazza.1762939@studenti.uniroma1.it
casadei.1952529@studenti.uniroma1.it

1 Introduction

In the field of Narrative Understanding and Storytelling, one of the most common issues is the lack of ad hoc data made up of stories with good narrative components that can be exploited for different types of tasks. In this work, we have built a custom dataset based on fantasy stories, which includes texts, summaries, and events.
We have identified and unified in the text the presence of the story's characters.
To demonstrate the validity of this work we have built and trained a graph neural network to classify the role of characters within the story.

2 Building the Dataset

In this project, one of the most interesting goals is to retrieve a novel type of data regarding narrative and storytelling, useful to accomplish different possible tasks.
As data source we chose the stories coming from *Critical Role*, (Fandom, 2023), a famous group of people who play *Dungeons & Dragons*, streaming the show. Upon this successful online content, a fanmade wiki has been created to store all the information on the events described in the game sessions.

2.1 Data extraction

So, the first step is focused on *web scraping* data from the roleplay game sessions in the Critical Role Fandom website, using the Python library *BeautifulSoup*. This library makes easier the navigation of HTML pages exploiting the tree structure, offering useful tools for the search and extraction of specific HTML tags.
This operation is a time-consuming task, considering the different structures of each page that have been taken into account. Moreover, these pages are continuously updated by the community, and sometimes fixing a version of the page, using web archives is indispensable. The scraped data is then processed to extract different kinds of information:

1. **Summaries**, text that summarizes what happens in each episode that is taken into account.
2. **Recap**, a small portion of text that describes the most important events coming from the previous episode.
3. **Timeline**, schematic description of major events of the story, sorted progressively for episode number in the campaign.
4. **Characters' role**, extracted all the characters retrieving the role, the aka list of names, the URL of the fandom page (used as a unique identifier for cited characters), and other minor information. Roles extracted are the following: Main, NPC (Non-playable character), Guest, ally, and antagonist. Has been created a different file for each narrative arc extracted, filtering for just characters belonging to the corresponding arc.
5. **Characters for episode**, list of characters contained in each episode, each character entry is enriched from the data coming from the files of characters' roles. This unification of data has been exploited using character web pages and simple regex patterns using the name and aka of characters.

Summaries and Recap data are first saved in a TSV file containing the plain text of the source and then several operations are performed like sentence boundary detection and tokenization, used to provide a corresponding id for each token. Furthermore, *Named Entity Recognition* and Fuzzy String Matching (Mouselimis, 2021) are used to identify and disambiguate character mentions in

the scraped data exploiting the information that the web page made available, such as hyperlinks to character pages and the various names by which they could be found. The results from these operations are saved as JSON as well as the timeline and story characters data.
The *NER* is performed with *Spacy*, using the performative pipeline that includes the transformer model with RoBERTa-base architecture (Liu et al., 2019).

The meaning of the characters' roles it's not obvious, the roles extracted from the wiki are related to the gaming context from which these stories are taken, namely the tabletop role-playing game. The "main" characters are those interpreted by a player, and are therefore what we would call the protagonists in a story; the "guest" characters are always played by a player, but they are guest stars of the referenced show, and will therefore be very present in that single game session, and then marginal in the rest of the story, a sort of co-protagonist for a limited narrative arc. The "NPCs", or Non-Player Characters, are the characters controlled by the Master and are therefore much more numerous than the "Main" characters. However, their presence within the story can be either extensive or minimal, they are the secondary characters. The final distinction between "ally" and "antagonist" is more moral than numerical, and is an extension of the "NPC" tag. is to indicate the helpers or adversaries of the story's heroes.

2.2 Pre-proccessing

After the extraction of the data from the source, before using all the information obtained, it's needed an intermediate process in order to make explicit the characters present in the various kind of data analyzed. The preliminary operation for doing this is to increment with an additional entry the list of characters for each episode, that describes the characters related to the main group in each episode. This is simply performed using the information already obtained.
In the consequent work, we still focus on the case of unnamed entities that refer to the main group, trying to find a list of substitutions, to have an explicit representation of the protagonists.
This list of substitutions has been carried out using string-matching patterns, POS tagging, and also considering the similarity between words. To measure this similarity, we have used *Glove* (Pennington et al., 2014) to obtain the word embedding, then the cosine distance has been computed looking for words (vectors) that have a very high cosine respect the target, described by several keywords which embody the main group.
After having overcome this problem we have still a lack of explicitness for single characters or groups that are not the protagonists. This is solved utilizing *Coreference resolution*.
This operation is performed using a very recent technology called *F-coref* (Otmazgin and Goldberg, 2022), which has been plugged directly into the spacy pipelines.
This model provides state-of-the-art coreference accuracy and at the same time is very fast. The architecture is based on the *LingMess* one and exploits an efficient batching implementation using a technique that the authors have called *Left-over batching*.
This model provides semantic clusters that are used to generate a new set of substitutions that are merged with the one already found and saved as JSON files.
The Final step is represented by the application of these substitutions in the original files of recap, summaries, and timeline. Also, the full list of characters representing the party has been directly inserted in the files.
This new version of data is now suitable to be processed in the creation of graph structures

2.3 Event Extraction

To extract the events from the pre-processed text, we used Verbatlas (Di Fabio et al., 2019). This is utilized to obtain, for each sentence containing at least two characters with different roles, the Verbatlas frame related to the actions, the list of characters involved, and their role. For each character, we also stored each sentence in which it appeared.

3 The model

Events and characters' information are fundamental data that is used to feed the model, in this section is described how information has been first, encoded and then used for the creation of the input graph. Finally are described the different types of models involved in the development. In the exposition that follows are utilized narrative arcs from the first campaign *"Vox Machina"* and the second campaign *"The Mighty Nein"*. The

training phase is based on the first 3 arcs of the first campaign, while the testing phase is on the first 2 arcs of the second campaign.

3.1 Feature engeneering

The objective of this sub-task is to pass from textual information describing characters or events to a numerical one, through the usage of a word embedding. To do this has been utilized the *Bert* model (Devlin et al., 2019), in two different manners depending on what to encode.
In the case of character feature encoding, all the sentences with relevant events containing the character are used, extracting the embedding in each sentence using the character's name and then doing the mean. Each single word embedding can be provided in 2 different ways, one that uses just the final output layer and another that sum results from the last 4 layers of the Bert Model.
In this way, each character is represented by a single vector of 768 values.
For what concert the events, the embedding is relative to the semantic description of *Verbatlas* frames, this is done by performing the sentence embedding which is exploited using the pooling layer of the model.

3.2 The Graphs

We use the NetworkX library to create two different graphs. The first graph has two types of nodes: character and event with the corresponding features mentioned above. Each Char node is only connected to Event nodes and thus the event representation extracted with Verbatlas is transposed into a graph form.
The second graph has only character nodes that are connected together whether involved in the same event present in the first type of graph.
Now the event features are in the edges that connect the char nodes to each other.

3.3 GNNs

For training, we used two models that we called "**ConvModel**" and "**SageModel**" which differ in the type of convolutional layer used. "ConvModel" uses GCNConv (Kipf and Welling, 2017) that computes new node representations as a linear combination of their neighbors' representations and their own representations. Instead, "SageModel" uses SAGEConv (Hamilton et al., 2018) that computes new node representations by aggregating their neighbors' representations using a specified aggregator (We have opted for the mean aggregator).
The rest of the model is almost identical in both cases, we start with a Dropout(0.5) (Srivastava et al., 2014) layer, a convolutional layer, Relu activation function (Fukushima, 1969), a BatchNormalization layer (Ioffe and Szegedy, 2015). These components are sequentially repeated another time using a different dropout probability of 0.3. In the end, we have a linear layer and the log softmax activation function (de Brébisson and Vincent, 2016) to receive the probability distribution.

3.3.1 Training

In the training phase, we used CrossEntropyLoss to compute the target. To overcome the problem of unbalanced classes we passed custom weights to the loss function, computed as the frequency of the class over the total samples. As optimizer, we used Adam (Kingma and Ba, 2014) with a learning rate of 0.02. Moreover, during the training we chose a variable number of epochs to accomplish several experiments, this number is around 2000-3000 epochs.

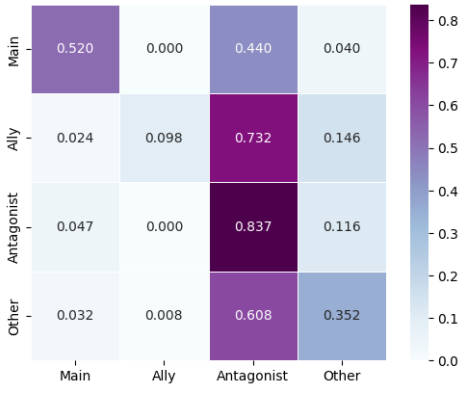
4 Experiments

In this section, we present the results obtained describing the experiments performed.

4.1 Recap Results

The first results that we propose come from the application of the discussed models on the recap data. In the following outcomes the label 'other' represents a character that is classified as "NPC" or "Guest".

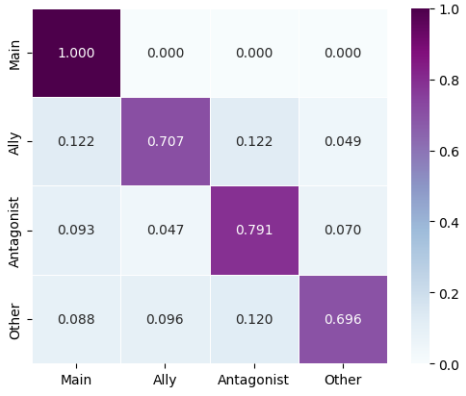
Results related to the execution of **ConvModel** on the Graph with only Characters.



Classes	Precision	Recall	F1-score	Support
Main	0.65	0.52	0.58	25
Ally	0.80	0.10	0.17	41
Antagonist	0.24	0.84	0.37	43
Other	0.79	0.35	0.49	125

	Precision	Recall	F1-score	Support
Accuracy			0.41	234
Macro avg	0.62	0.45	0.40	234
Weighted avg	0.67	0.41	0.42	234

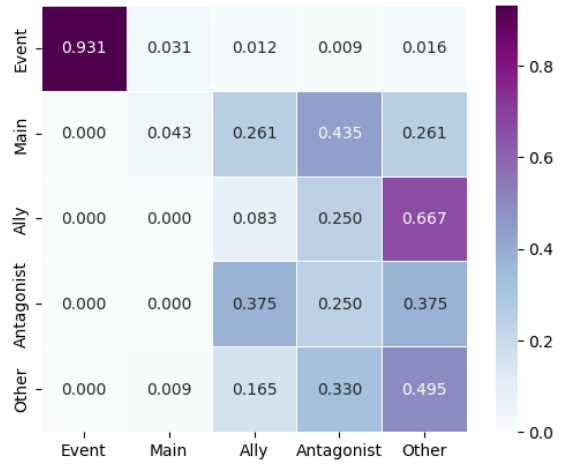
Results related to the execution of **SageModel** on the Graph with only Characters.



Classes	Precision	Recall	F1-score	Support
Main	0.56	1.00	0.71	25
Ally	0.67	0.71	0.69	41
Antagonist	0.63	0.79	0.70	43
Other	0.95	0.70	0.80	125

	Precision	Recall	F1-score	Support
Accuracy			0.75	234
Macro average	0.70	0.80	0.73	234
Weighted average	0.80	0.75	0.75	234

Results related to the execution of **SageModel** on the Graph containing Events and Characters.



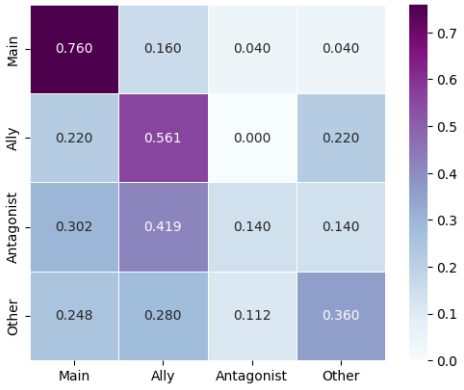
Classes	Precision	Recall	F1-score	Support
Event	1.00	0.93	0.96	3823
Main	0.01	0.04	0.01	23
Ally	0.01	0.08	0.02	12
Antagonist	0.02	0.25	0.04	8
Other	0.40	0.50	0.44	109

	Precision	Recall	F1-score	Support
Accuracy			0.91	3975
Macro average	0.29	0.36	0.30	3975
Weighted average	0.97	0.91	0.94	3975

4.2 Summaries Results

Now we show the test outcomes obtained using the data coming from the summaries.

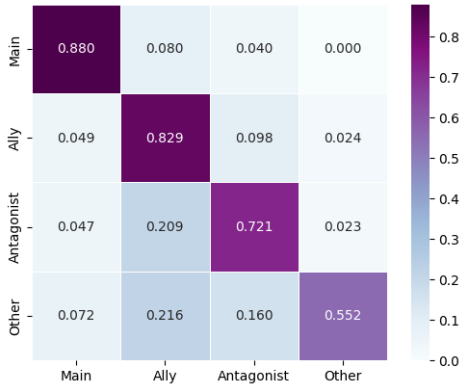
Results related to the execution of **ConvModel** on the Graph with only Characters



Classes	Precision	Recall	F1-score	Support
Main	0.26	0.76	0.39	25
Ally	0.29	0.56	0.38	41
Antagonist	0.29	0.14	0.19	43
Other	0.74	0.36	0.48	125

	Precision	Recall	F1-score	Support
Accuracy			0.40	234
Macro average	0.39	0.46	0.36	234
Weighted average	0.53	0.40	0.40	234

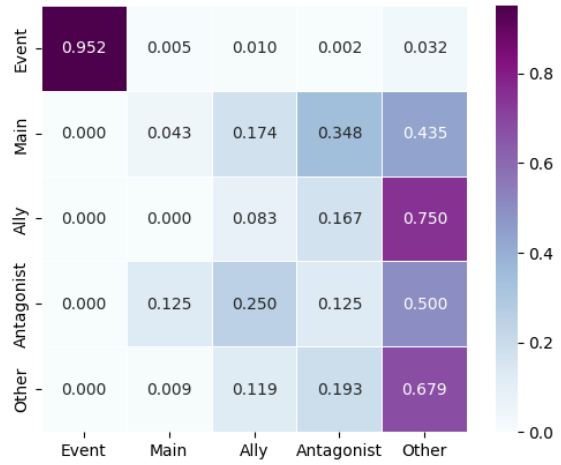
Results related to the execution of **SageModel** on the Graph with only Characters



Classes	Precision	Recall	F1-score	Support
Main	0.63	0.88	0.73	25
Ally	0.47	0.83	0.60	41
Antagonist	0.55	0.72	0.63	43
Other	0.97	0.55	0.70	125

	Precision	Recall	F1-score	Support
Accuracy			0.67	234
Macro average	0.66	0.75	0.67	234
Weighted average	0.77	0.67	0.67	234

Results related to the execution of **SageModel** on the Graph containing Events and Characters.



Classes	Precision	Recall	F1-score	Support
Event	1.00	0.95	0.98	3823
Main	0.04	0.04	0.04	23
Ally	0.02	0.08	0.03	12
Antagonist	0.03	0.12	0.04	8
Other	0.34	0.68	0.45	109

	Precision	Recall	F1-score	Support
Accuracy			0.93	3975
Macro average	0.29	0.38	0.31	3975
Weighted average	0.97	0.93	0.95	3975

5 Conclusion

The results obtained are very varied, the use of the **SageModel** gives the best performance when used on the graph containing only the characters, whereas when we tested the graph with events and characters we encountered serious problems in training the network due mainly to the unbalanced dataset (despite the fact that we tried to introduce class weights as explained above). With regard to the **ConvModel**, although it succeeds in learning some of the characteristics that allow the classification of roles, it doesn't, however, reach the level of the **SageModel** and was therefore included only for comparison.
This novel project emphasizes how the task of building a new dataset from scratch can be hard. Has been shown a complete pipeline that first acquires the data, performs the pre-processing, builds the input graph, defines and uses the model, and finally makes an evaluation. Furthermore, The proposed task is not trivial, the association of each character to the corresponding label is not always objective, given the possibility to have more than one label that are correctly related to a character.

Possible further implementations are possible to continue working with this data, exploring new possible tasks related the narrative understanding, i.e. emotional arcs classification.
in addition, there is the possibility to enhance and expand the dataset, including more content from Critical Role and other similar sources. An interesting path to follow could be the extension to the multilingual domain, in fact, there are many wikis like the one used to create this dataset and in many languages, so one could exploit this type of data as well.

References

Alexandre de Brébisson and Pascal Vincent. 2016. An exploration of softmax alternatives belonging to the spherical loss family.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Andrea Di Fabio, Simone Conia, and Roberto Navigli. 2019. Verbatlas: a novel large-scale verbal semantic resource and its application to semantic role labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th*

International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 627–637.

Fandom. 2023. Critical role wiki.

Kunihiko Fukushima. 1969. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2018. Inductive representation learning on large graphs.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Lampros Mouselimis. 2021. *fuzzywuzzyR: Fuzzy String Matching*. R package version 1.0.5.

Arie Cattan Otmazgin, Shon and Yoav Goldberg. 2022. F-coref: Fast, accurate and easy to use coreference resolution. *arXiv preprint arXiv:2209.04280*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.